

A large, stylized letter 'A' is formed using the characters 'S' and 'Y'. The 'S' characters are arranged in a grid-like pattern to form the left and right sides of the letter, while 'Y' characters form the central vertical stem and the horizontal crossbar. The overall shape is a bold, blocky 'A' that fills most of the page.

LO  
VO

```

LL               IIIIII               SSSSSSSS
LL               IIIIII               SSSSSSSS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SSSSSS
LL               II                   SSSSSS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SSSSSS
LLLLLLLLLLLLLL IIIIIIII             SSSSSSSS
LLLLLLLLLLLLLL IIIIIIII             SSSSSSSS

```



(2) 63  
(3) 121  
(4) 330  
(5) 467  
(6) 561

DECLARATIONS  
EXES\$LOAD\_CODE - Perform Actual Code Load  
EXES\$LOAD\_NONPAGD - Load code into non paged memory  
EXES\$LOAD\_PAGED - Load code into paged memory  
EXES\$SYS\_SECTION - Create a system section

```
0000 1 .TITLE LOADSUB - System Code Loading Subroutines
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27
0000 28 ++
0000 29 Facility: System Code Loader
0000 30
0000 31 Abstract: These routines performs operations to load code into
0000 32 system address space.
0000 33
0000 34 Environment: Kernel Mode.
0000 35
0000 36 Author: Jeffrey W. Horn, Creation Date: 1-MAR-1983
0000 37
0000 38 Modified by:
0000 39
0000 40 V03-007 WMC0007 Wayne Cardoza 12-Jan-1983
0000 41 SYSWRTABL should make sections CRF.
0000 42
0000 43 V03-006 WMC0006 Wayne Cardoza 05-Jan-1984
0000 44 Make sure SPTs released on error.
0000 45
0000 46 V03-005 WMC0005 Wayne Cardoza 09-Dec-1983
0000 47 Enable all ISD checking.
0000 48
0000 49 V03-004 WMC0004 Wayne Cardoza 07-Sep-1983
0000 50 Make sure we never use SYSGEN private copy of cells.
0000 51
0000 52 V03-003 WMC0003 Wayne Cardoza 29-Jul-1983
0000 53 More of the same.
0000 54
0000 55 V03-002 WMC0002 Wayne Cardoza 24-Jun-1983
0000 56 Fix assorted bugs after testing.
0000 57
```



- System Code Loading Subroutines<sup>L 3</sup>

Page 2  
(1)

```

0000      58 :
0000      59 :
0000      60 :
0000      61 :--

```

V03-001 WMC0001  
Complete it.

Wayne Cardoza 24-May-1983

[illegible]

```
0000 63      .SBTTL  DECLARATIONS
0000 64
0000 65      :
0000 66      : Include files:
0000 67      :
0000 68      :
0000 69      :
0000 70      : Macros:
0000 71      :
0000 72      :
0000 73      $CCBDEF
0000 74      $DPTDEF
0000 75      $DYNDEF
0000 76      $IHDDEF
0000 77      $IODEF
0000 78      $IPLDEF
0000 79      $ISDDEF
0000 80      $PHDDEF
0000 81      $PFNDEF
0000 82      $PRTDEF
0000 83      $PTEDEF
0000 84      $SECDEF
0000 85      $SLVDEF
0000 86      $SSDEF
0000 87      $VADEF
0000 88      $WCBDEF
0000 89
0000 90      :
0000 91      : Equated Symbols:
0000 92      :
0000 93      :
0000 94      :
0000 95      : Own Storage
0000 96      :
0000 97      :
0000 98      : This table is used to map the page protection codes into
0000 99      : codes which allow at least Kernel Mode writeability
0000 100
00 0000 101 KW_TBL: .BYTE  PRTSC_NA      : NA => NA
01 0001 102      .BYTE  PRTSC_RESERVED : RESERVED => RESERVED
02 0002 103      .BYTE  PRTSC_KW      : KW => KW
02 0003 104      .BYTE  PRTSC_KW      : KR => KW
04 0004 105      .BYTE  PRTSC_UW      : UW => UW
05 0005 106      .BYTE  PRTSC_EW      : EW => EW
06 0006 107      .BYTE  PRTSC_ERKW     : ERKW => ERKW
06 0007 108      .BYTE  PRTSC_ERKW     : ER => ERKW
08 0008 109      .BYTE  PRTSC_SW      : SW => SW
09 0009 110      .BYTE  PRTSC_SREW     : SREW => SREW
0A 000A 111      .BYTE  PRTSC_SRKW     : SRKW => SRKW
0A 000B 112      .BYTE  PRTSC_SRKW     : SR => SRKW
0C 000C 113      .BYTE  PRTSC_URSW     : URSW => URSW
0D 000D 114      .BYTE  PRTSC_UREW     : UREW => UREW
0E 000E 115      .BYTE  PRTSC_URKW     : URKW => URKW
0E 000F 116      .BYTE  PRTSC_URKW     : UR => URKW
      0010 117
00000418 0010 118 STORAGE:
      0010 119      .BLKB  ^X408
```



```
0418 121      .SBTTL  EXE$LOAD_CODE - Perform Actual Code Load
0418 122
0418 123      :++
0418 124      : EXE$LOAD_CODE - Perform Actual Code Load
0418 125
0418 126      : This subroutine makes code resident in system space, either
0418 127      : as a new system section for pageable code, or read into
0418 128      : created system virtual address space for nonpageable code.
0418 129
0418 130      : Calling Sequence:
0418 131
0418 132      : CALLS  #2,EXE$LOAD_CODE
0418 133
0418 134      : Input Parameters: (AP offset)
0418 135
00000004 0418 136      : CHAN  =      4      ; channel file is accessed on
0418 137
0418 138      : Implicit Inputs:
0418 139      : Contents of the image file including Image Header and Prologue in
0418 140      : first block of image.
0418 141
0418 142      : Output Parameters: (AP offset)
0418 143
0418 144      : R0      =      Completion Status
00000008 0418 145      : RETADR  =      8      ; address to return starting VA
0418 146
0418 147      : Implicit Outputs:
0418 148      : None.
0418 149
0418 150      : Side Effects:
0418 151
0418 152      :--
0418 153
00FC 0418 154      .ENTRY  EXE$LOAD_CODE,*M<R2,R3,R4,R5,R6,R7>
041A 155      .ENABL  LSB
041A 156      : MOVAB  -<<2*^X200>+8>(SP),SP      ; allocate scratch space
041A 157      : MOVL   SP,R3      ; save scratch space address
53   FBF2 CF  9E 041A 158      MOVAB  STORAGE,R3
041F 159
041F 160      : Read in image header into buffer
041F 161
041F 162
56   0400 C3  DE 041F 163      MOVAL   ^X400(R3),R6      ; save addr of IOSB
0424 164
0424 165      $QIOW_S EFN  = #1,-
0424 166      CHAN  = CHAN(AP),-
0424 167      FUNC  = #IOS_READVBLK,-
0424 168      IOSB  = (R6),-
0424 169      P1    = (R3),-
0424 170      P2    = #512,-
0424 171      P3    = #1
01 50  E8 0446 172      BLBS   R0,20$      ; get out on error
50   66  3C 0449 173 10$: RET
F9 50  E9 044A 174 20$: MOVZWL (R6),R0      ; get IOSB status
044D 175      BLBC   R0,10$      ; get out on error
0450 176
0450 177 ;
```



```
0450 178 : Now read in first page of image
0450 179 :
0450 180 :
54 0200 C3 DE 0450 181 MOVAL ^X200(R3),R4 ; get addr of image buffer
51 10 A3 9A 0455 182 MOVZBL IHD$B_HDRBLKCNT(R3),R1 ; get number blocks in header
51 D6 0459 183 INCL R1 ; one past is first block of image (P3)
045B 184 $QIOW_S EFN = #1,-
045B 185 CHAN = CHAN(AP),-
045B 186 FUNC = #IOS_READVBLK,-
045B 187 IOSB = (R6),-
045B 188 P1 = (R4),-
045B 189 P2 = #512,-
045B 190 P3 = R1
C9 50 E9 047D 191 BLBC R0,10$ ; get out on error
50 66 3C 0480 192 MOVZWL (R6),R0 ; get IOSB status
C3 50 E9 0483 193 BLBC R0,10$ ; get out on error
0486 194
52 50 63 3C 0486 195 MOVZWL (R3),R0 ; offset to ISD's
53 50 C1 0489 196 ADDL3 R0,R3,R2
048D 197
048D 198 :
048D 199 :
048D 200 :
048D 201 :
56 56 08 A4 3C 048D 202 MOVZWL SLV$W_SIZE(R4),R6 ; code size
56 000001FF 8F C0 0491 203 ADDL #511,R6 ; get a page count
56 56 F7 8F 78 0498 204 ASHL #-9,R6,R6
049D 205 DSBINT 25$ ; go to SYNCH and lock down code
55 00000000'EF 9E 04A7 206 MOVAB MMG$A_SYSPARAM,R5 ; make sure we don't use SYSGEN private copy
57 0000'C5 D0 04AE 207 MOVL BOO$GL_SPTFREL-EXESA_SYSPARAM(R5),R7 ; first free SPT
50 56 57 C1 04B3 208 ADDL3 R7,R6,R0 ; new free pointer
0000'C5 50 D1 04B7 209 CMPL R0,BOO$GL_SPTFREL-EXESA_SYSPARAM(R5) ; enough left?
0A 14 04BC 210 BGTR NOSPT ; branch if not
0000'C5 50 D0 04BE 211 MOVL R0,BOO$GL_SPTFREL-EXESA_SYSPARAM(R5) ; record the allocation
04C3 212 ENBINT
11 11 04C6 213 BRB 26$
04C8 214
04C8 215 NOSPT: ENBINT
50 0244 8F 3C 04CB 216 MOVZWL #SS$_VASFULL,R0
04 04D0 217 RET
04D1 218
00000008 04D1 219 25$: .LONG IPL$_SYNCH
04D5 220
50 14 D0 04D5 221 ERRTP: MOVL #SS$_BADPARAM,R0
04 04D8 222 RET
04D9 223 26$:
56 02 A2 A2 04D9 224 SUBW ISD$W_PAGCNT(R2),R6 ; SPT left after first ISD
03 18 04DD 225 BGEQ 27$ ; done here so driver load works
00A3 31 04DF 226 BRW BADHDR ; image header doesn't match SLV
08 BC 57 09 78 04E2 227 27$: ASHL #9,R7,@RETADR(AP) ; return address of loaded code
00 08 BC 1F E2 04E7 228 BBSS #V$V_SYSTEM,@RETADR(AP),28$
04EC 229 28$:
04EC 230 :
04EC 231 :
04EC 232 :
7E 57 DD 04EC 233
0E 0E 9A 04EE 234 Set up initial parameters for load routines
PUSHL R7 ; first SPT
MOVZBL #PRT$_URKW,-(SP) ; driver protection
```



```
7E 02 A2 3C 04F1 235 MOVZWL ISD$W_PAGCNT(R2),-(SP) ; page count
7E 10 A3 9A 04F5 236 MOVZBL IHD$B_HDRBLKCNT(R3),-(SP) ; start VBN = one past image header
    6E D6 04F9 237 INCL (SP) ;
    04 AC DD 04FB 238 PUSHL CHAN(AP) ; channel
    05 DD 04FE 239 PUSHL #5 ; argument count
    0500 240
    0500 241
    0500 242
    0500 243
    0500 244 Drivers get special treatment - single ISD
1E 0A A4 91 0500 245 CMPB SLV$B_TYPE(R4),#DYN$C_DPT ; see if driver
    0E 12 0504 246 BNEQ 35$ ; branch if not
00000609'EF 6E FA 0506 247 CALLG (SP),EXES$LOAD_NONPAGD ; go load non paged code
    03 50 E8 050D 248 BLBS R0,30$
    0079 31 0510 249 BRW NONPAG_ERR
    04 0513 250 30$: RET
    0514 251
    62 8F 0A A4 91 0514 252 35$: CMPB SLV$B_TYPE(R4),#DYN$C_LOADCODE ; see if loadable code
    BA 12 0519 253 BNEQ ERRTP ; error if not
08 A2 00060405 8F D3 051B 254 40$: BITL #ISD$M_DZRO ! ISD$M_VECTOR -
    0523 255 ! ISD$M_GBL ! ISD$M_FIXUPVEC -
    0523 256 ! ISD$M_PROTECT,ISD$M_FLAGS(R2)
    60 12 0523 257 BNEQ BADHDR ; illegal ISD types
    0525 258
    0525 259 Set up argument list for loader for next ISD
    0525 260
0008'CE 0C A2 D0 0525 261 MOVL ISD$M_VBN(R2),STRIVBN(SP) ; starting image VBN
000C'CE 02 A2 3C 052B 262 MOVZWL ISD$W_PAGCNT(R2),PAGECNT(SP) ; ISD pagecount
    07 08 A2 03 E0 0531 263 BBS #ISD$M_WRT,ISD$M_FLAGS(R2),50$ ; is it writeable
    10'AE 0C A4 9A 0536 264 MOVZBL SLV$B_PROT_R(R4),B^PROT(SP) ; get read-only page protection
    05 11 053B 265 BRB 60$
    10'AE 0D A4 9A 053D 266 50$: MOVZBL SLV$B_PROT_W(R4),B^PROT(SP) ; get writeable page protection
    0542 267
    0542 268 Pick the correct loader routine
    0542 269
06 00000000'9F 00' E1 0542 270 60$: BBC S^#EXES$V_SYSPAGING,#EXES$GL_FLAGS,70$ ; branch if not paging
    02 0B A4 91 054A 271 CMPB SLV$B_SUBTYP(R4),#DYN$C_PAGED ; see if pageable
    0C 13 054E 272 BEQL 80$ ; branch if so
    0550 273
00000609'EF 6E FA 0550 274 70$: CALLG (SP),EXES$LOAD_NONPAGD ; go load non paged code
    32 50 E9 0557 275 BLBC R0,NONPAG_ERR
    0A 11 055A 276 BRB 90$
    055C 277
000006FA'EF 6E FA 055C 278 80$: CALLG (SP),EXES$LOAD_PAGED ; go map paged code
    1E 50 E9 0563 279 BLBC R0,110$ ; must be no STX, don't try to clean up mess
    0566 280
    0566 281 Next ISD
    0566 282
    53 62 3C 0566 283 90$: MOVZWL ISD$W_SIZE(R2),R3
    52 53 C0 0569 284 ADDL R3,R2 ; next ISD
    62 B5 056C 285 TSTW ISD$W_SIZE(R2) ; are we done
    11 13 056E 286 BEQL 100$
    13 19 0570 287 BLSS BADHDR ; error - there can't be this many ISD's
    56 02 A2 A2 0572 288 SUBW ISD$W_PAGCNT(R2),R6
    0D 19 0576 289 BLSS BADHDR ; too many pages in the image
0014'CE 000C'CE C0 0578 290 ADDL PAGECNT(SP),SPT(SP) ; next SPT
    9A 11 057F 291 BRB 40$ ; process next ISD
```



```

      50 01 3C 0581 292
      04 0581 293 100$: MOVZWL #SS$_NORMAL,R0
      0584 294 110$: RET
      0585 295
      0585 296 .DSABL LSB
      0585 297
50 00000044 8F D0 0585 298 BADHDR: MOVL #SS$_BADIMGHDR,R0
      058C 299
      058C 300 NONPAG_ERR:
07 00000000'9F 00' E1 058C 301 BBC S^#EXES$V SYSPAGING,@#EXES$GL FLAGS, 5$ ; branch if not paging
      02 0B A4 91 0594 302 CMPB SLV$B_SUBTYP(R4),#DYN$C_PAGED ; see if pageable
      01 12 0598 303 BNEQ 5$ ; branch if so
      04 059A 304 RET ; don't try to clean up if paged
      50 DD 059B 305 5$: PUSHL R0 ; save error status
      56 08 A4 3C 059D 306 MOVZWL SLV$W_SIZE(R4),R6 ; code size
56 000001FF 8F C0 05A1 307 ADDL #511,R6 ; get a page count
56 56 F7 8F 78 05A8 308 ASHL #-9,R6,R6
59 57 56 C1 05AD 309 ADDL3 R6,R7,R9 ; save next SPT after ours
      05B1 310 DSBINT 50$ ; go to SYNCH and lock down code
58 00000000'FF47 DE 05BB 311 MOVAL @MMG$GL_SPTBASE[R7],R8 ; SVAPTE of first SPT
50 68 15 00 EF 05C3 312 10$: EXTZV #PTES$V_PFN,#PTES$P_PFN,(R8),R0 ; get the PFN
      21 13 05C8 313 BEQL 20$ ; done
      00000000'FF40 D4 05CA 314 CLRL @PFN$AL_PTE[R0] ; clear back pointer
      C0000000'FF40 B7 05D1 315 DECW @PFN$AW_REFCNT[R0] ; decrement the ref count
      24 12 05D8 316 BNEQ 30$ ; some one knows about it - give up
      00000000'EF 16 05DA 317 JSB MMG$DALLOCPFN ; free PFN
      88 D4 05E0 318 CLRL (R8)+ ; invalidate PTE
      00000000'EF D6 05E2 319 INCL PFN$GL_PHYPGCNT ; count the freed page
      D8 56 F5 05E8 320 SOBGTR R6,10$
50 00000000'EF 9E 05EB 321 20$: MOVAB MMG$A_SYSPARAM,R0 ; make sure we don't use SYSGEN private copy
      0000'C0 59 D1 05F2 322 CMPL R9,B00$GL_SPTFREL-EXES$A_SYSPARAM(R0) ; can we give back the SPTs
      05 12 05F7 323 BNEQ 30$ ; no - more have been allocated
      0000'C0 57 D0 05F9 324 MOVL R7,B00$GL_SPTFREL-EXES$A_SYSPARAM(R0) ; reset free SPT pointer
      05FE 325 30$: ENBINT
      50 8E D0 0601 326 MOVL (SP)+,R0
      04 0604 327 RET
      00000008 0605 328 50$: .LONG IPL$_SYNCH
```



```
0609 330 .SBTTL EXE$LOAD_NONPAGD - Load code into non paged memory
0609 331
0609 332 :++
0609 333 : LOAD_NON_PAGED - Load code into non paged memory
0609 334 :
0609 335 : This routine loads code into non paged memory using the following
0609 336 : algorithm:
0609 337 :
0609 338 : 1. For each SPT:
0609 339 : a. Allocate a physical page
0609 340 : b. Fill in PFN data-base
0609 341 : c. Fill in SPT.
0609 342 : 2. Read in code into new address space
0609 343 : 3. Set page protection on new address space.
0609 344 :
0609 345 : Page protection is the PROT(AP) value unless page is first page in
0609 346 : image or the WRITEABLESYS parameter is set, then the protection is
0609 347 : translated into one which allows at least kernel mode write.
0609 348 :
0609 349 : Calling Sequence:
0609 350 :
0609 351 : CALLS #5,EXE$LOAD_NONPAGD
0609 352 :
0609 353 : Input Parameters: (AP offsets)
0609 354 :
00000004 0609 355 : CHAN = 4 ; Channel file is access on
00000008 0609 356 : STRTVBN = 8 ; image start VBN
0000000C 0609 357 : PAGECNT = 12 ; number of pages to be loaded
00000010 0609 358 : PROT = 16 ; protection to be applied
00000014 0609 359 : SPT = 20 ; first SPT index
0609 360 :
0609 361 : Implicit Inputs:
0609 362 :
0609 363 : Output Parameters:
0609 364 :
0609 365 : Implicit Outputs:
0609 366 :
0609 367 : Completion Codes:
0609 368 :
0609 369 : Side Effects:
0609 370 :
0609 371 :--
0609 372 :
07FC 0609 373 :.ENTRY EXE$LOAD_NONPAGD,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
0609 374 :
54 14 AC 09 78 0609 375 ASHL #9,SPT(AP),R4 ; compute VA of assigned SPT
00 54 1F E2 0609 376 BBSS #VASV_SYSTEM,R4,10$ ; set system bit
0614 377 :
0614 378 : Now fill in those PTEs
0614 379 :
0614 380 :
0614 381 : Set up loop
0614 382 :
58 57 14 AC D0 0614 383 10$: MOVL SPT(AP),R7 ; get initial SPT index
00000000 FF47 DE 0618 384 MOVAL @MMG$GL_SPTBASE[R7],R8 ; get SVAPTE of first PTE
59 0C AC 01 C3 0620 385 SUBL3 #1,PAGECNT(AP),R9 ; get ending index
0625 386
```



```
0625 387 :  
0625 388 :  
0625 389 :  
0625 390 :  
0625 391 :  
0625 392 :  
062F 393 20$: DSBINT 40$ ; go to SYNCH and lock down code  
00000000'9F 16 062F 394 JSB @#MMG$ALLOC PFN ; attempt to allocate a PFN  
50 D5 0635 395 TSTL R0 ; check status  
3A 19 0637 396 BLSS 30$ ; branch if no page allocated  
0639 397  
0639 398 :  
0639 399 :  
00000000'FF40 B6 0639 400 INCW @PFNS$AW REFCNT[R0] ; set reference count  
00000000'FF40 6849 DE 0640 401 MOVAL (R8)[R9],@PFNS$AL PTE[R0] ; set SVAPTE in PTE back pointer  
00000000'FF40 07 90 0649 402 MOV B #PFNS$C ACTIVE,@PFNS$AB_STATE[R0] ; set state as active  
00000000'FF40 01 90 0651 403 MOV B #1,@PFNS$AB_TYPE[R0] ; set type as system page  
00000000'9F D7 0659 404 DECL @#PFNS$GL PHYPGCNT ; one less physical page  
50 90000000 8F C9 065F 405 BISL3 #<PTES$M_VALID!PTES$C_KW>,-  
6849 0666 406 R0,(R8)[R9] ; set valid prot, PFN into PTE  
0668 407  
0668 408 :  
0668 409 :  
C4 59 F4 0668 410 SOBGEQ R9,20$  
0668 411 INVALID  
066E 412 ENBINT ; finished with memory man.  
10 11 0671 413 BRB READCOD  
0673 414  
50 0124 8F 3C 0673 415 30$: MOVZWL #SS$_INSFMEM,R0  
0678 416 INVALID  
067B 417 ENBINT  
04 067E 418 RET  
067F 419  
00000008 067F 420 40$: .LONG IPL$_SYNCH  
0683 421  
0683 422 :  
0683 423 :  
0683 424 :  
0683 425 :  
0683 426 READCOD:  
0683 427  
50 52 7E 7C 0683 428 CLRQ -(SP) ; allocate IOSB  
50 0C AC 5E D0 0685 429 MOVL SP,R2 ; get IOSB address  
0688 430 ASHL #9,PAGECNT(AP),R0 ; byte count (P2)  
068D 431 $QIOW_S EFN = #1,-  
068D 432 CHAN = CHAN(AP),-  
068D 433 FUNC = #IOS_READVBLK,-  
068D 434 IOSB = (R2),-  
068D 435 P1 = (R4),-  
068D 436 P2 = R0,-  
068D 437 P3 = STRTVBN(AP)  
01 50 E8 06AC 438 BLBS R0,6$  
50 62 04 06AF 439 5$: RET ; get out on error  
F9 50 E9 06B0 440 6$: MOVZWL (R2),R0 ; get iosb status  
5E 08 C0 06B6 441 BLBC R0,5$  
06B6 442  
06B6 443 ADDL #8,SP ; deallocate IOSB
```



```

06B9 444 :
06B9 445 :
06B9 446 :
06B9 447 :
06B9 448 :
59 0C AC 01 C3 06C3 449 DSBINT 20$ ; get to SYNCH and lock down code
53 10 AC D0 06C8 450 SUBL3 #1,PAGECNT(AP),R9 ; get index of last PTE
00000000'9F 00' E1 06CC 451 MOVL PROT(AP),R3 ; get protection code
53 F927 CF43 9A 06D3 452 BBC S^#EXESV_SYSWRTABL, - ; branch if no WRITEABLESYS
50 50 6849 DE 06D4 453 MOVZBL KW TBL[R3],R3 ; change prot to at least kern write
60 04 1B 53 F0 06DA 454 10$: MOVAL (R8)[R9],R0 ; get SVAPTE of page
F4 59 F4 06DE 455 INSV R3,#PTESV_PROT, - ; set page protection
06E3 456 SOBGEQ R9,10$
06E6 458
06E6 459 INVALID
06E9 460 ENBINT
06EC 461
50 01 D0 06EC 462 MOVL #SS$_NORMAL,R0
04 06EF 463 RET
00000008 06F0 464 20$: .LONG IPL$_SYNCH
06F4 465
```



```
06F4 467 .SBTTL EXES$LOAD_PAGED - Load code into paged memory
06F4 468
06F4 469 :++
06F4 470 : EXES$LOAD_PAGED - Load code into paged memory
06F4 471 :
06F4 472 : This routine does not actually load the code into System Paged Memory
06F4 473 : but instead sets up a System Section which can then be paged in.
06F4 474 :
06F4 475 : Page protection is the PROT(AP) value unless page is first page in
06F4 476 : image or the WRITEABLESYS parameter is set, then the protection is
06F4 477 : translated into one which allows at least kernel mode write.
06F4 478 :
06F4 479 : Calling Sequence:
06F4 480 :
06F4 481 : CALLS #5,EXES$LOAD_PAGED
06F4 482 :
06F4 483 : Input Parameters: (AP offsets)
06F4 484 :
06F4 485 :         CHAN      =      4           ; channel file is accessed on
06F4 486 :                                     ; or WCB address
06F4 487 :         STRTVBN   =      8           ; image start VBN
06F4 488 :         PAGECNT   =     12           ; number of pages to be loaded
06F4 489 :         PROT      =     16           ; protection to be applied
06F4 490 :         SPT       =     20           ; index of first SPT
06F4 491 :
06F4 492 : Implicit Inputs:
06F4 493 :
06F4 494 : Output Parameters:
06F4 495 :
06F4 496 : Implicit Outputs:
06F4 497 :
06F4 498 : Completion Codes:
06F4 499 :
06F4 500 : Side Effects:
06F4 501 :
06F4 502 :--
06F4 503
50 026C 8F 3C 06F4 504 ERRCHN: MOVZWL #SS$_IVCHNLSEC,R0
06F4 505 RET
06FA 506
07FC 06FA 507 .ENTRY EXES$LOAD_PAGED,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
50 04 AC D0 06FC 508 MOVL CHAN(AP),R0 ; get channel number
54 50 D0 0700 509 MOVL R0,R4
00000000'9F 16 0703 510 BLSS 20$
54 E6 50 E9 0705 511 JSB @#IOC$VERIFYCHAN ; get CCB address
05 0B A4 03 E2 070B 512 BLBC R0,ERRCHN
0712 513 MOVL CCB$_WIND(R1),R4 ; get WCB address
0717 514 10$: BBSS #WCB$_SHRWCB,WCB$_ACCESS(R4),20$ ; set share bit of WCB
0717 515 ; branch if already set
0717 516
0717 517 ASSUME WCB$_REFCNT EQ WCB$_PID+2
0C A4 01 10 9C 0717 518
071C 519 ROTL #16,#1,WCB$_PID(R4) ; make PID invalid, refcount = 1
071C 520
55 00000000'9F D0 0722 521 20$: DSBINT #IPL$_ASTDEL ; don't let process be deleted
00000000'9F 16 0729 522 MOVL @#MMG$GL_SYSPHD,R5 ; get address of system head
0729 523 JSB @#MMG$ALCSTX ; allocate a system section
```



```
58 55 75 50 E9 072F 524 BLBC R0,60$ ; get out on error
      20 A5 C1 0732 525 ADDL3 PHD$PSTBASOFF(R5),R5,R8 ; base address of section table
      58 6841 DE 0737 526 MOVAL (R8)[R1],R8 ; address of section table entry
      68 D4 073B 527 CLRL SEC$LCB(R8) ; no channel control block address
      OC A8 54 D0 073D 528 MOVL R4,SEC$WINDOW(R8) ; set window control block address
10 A8 08 AC D0 0741 529 MOVL STRTVBN(AP),SEC$VBN(R8) ; start VBN of section
      57 14 AC D0 0746 530 MOVL SPT(AP),R7
      08 A8 57 D0 074A 531 MOVL R7,SEC$VPXPFC(R8) ; starting SPT index
      14 A8 B4 074E 532 CLRW SEC$W_FLAGS(R8) ; zero section flags
1C A8 0C AC D0 0751 533 MOVL PAGECNT(AP),SEC$PAGECNT(R8) ; size of section in pages
18 A8 0C AC D0 0756 534 MOVL PAGECNT(AP),SEC$REFCNT(R8) ; number of outstanding references
      04 A8 D4 075B 535 CLRL SEC$W_SECXFL(R8) ; no section indices
      53 10 AC D0 075E 536
00000000'9F 00' E1 0762 537 MOVL PROT(AP),R3 ; get protection code
      53 F891 CF43 9A 076A 539 BBC S^#EXES$SYSWRITABLE, - ; branch if no WRITEABLESYS
      14 A8 02 A8 0770 540 MOVZBL KW TBL[R3],R3 ; change prot to at least kern write
51 10 10 00000440 8F F0 0774 541 BISW #SEC$M_CRF,SEC$W_FLAGS(R8) ; make it CRF
      077D 542 30$: INSV #<PTE$M_TYP1!PTE$M_TYPO>@-16, -
      077D 543 #16,#16,R1 ; form section type pte
51 04 1B 53 F0 077D 544 INSV R3,#PTE$V_PROT,#PTE$S_PROT,R1 ; set prot code into pte
58 00000000'9F D0 0782 545 MOVL @#MMG$GL_SPTBASE,R8 ; get SPT base address
      58 6847 DE 0789 546 MOVAL (R8)[R7],R8 ; get SVAPTE of first PTE
59 0C AC 01 C3 078D 547 SUBL3 #1,PAGECNT(AP),R9 ; get index of last PTE
      0792 548
      6849 51 D0 0792 549 40$: MOVL R1,(R8)[R9] ; set PTE contents
      F9 59 F4 0796 550 SOBGEQ R9,40$
      0799 551
03 0B A4 03 E1 0799 552 BBC #WCB$V_SHRWCW,WCB$B_ACCESS(R4),50$ ; branch if not shared WCB
      0E A4 B6 079E 553 INCW WCB$W_REFCNT(R4) ; count another pointer to WCB
      07A1 554
      07A1 555 50$: INVALID
      07A4 556 MOVZBL #SS$NORMAL,R0
      07A7 557
      07A7 558 60$: ENBINT
      04 07AA 559 RET
```



```
07AB 561 .SBTTL EXESSYS_SECTION - Create a system section
07AB 562
07AB 563 :++
07AB 564 : EXESSYS_SECTION - Create a system section
07AB 565 :
07AB 566 : This routine will create a system section.
07AB 567 :
07AB 568 : Page protection is the PROT(AP) value unless page is first page in
07AB 569 : image or the WRITEABLESYS parameter is set, then the protection is
07AB 570 : translated into one which allows at least kernel mode write.
07AB 571 :
07AB 572 : Calling Sequence:
07AB 573 :
07AB 574 : CALLS #5,EXESSYS_SECTION
07AB 575 :
07AB 576 : Input Parameters: (AP offsets)
07AB 577 :
00000004 07AB 578 : SECCHAN = 4 ; Channel file is access on
00000008 07AB 579 : SECSTRTVBN= 8 ; image start VBN
0000000C 07AB 580 : SECPAGECNT = 12 ; number of pages to be loaded
00000010 07AB 581 : SECPROT = 16 ; protection to be applied
07AB 582 :
07AB 583 : Implicit Inputs:
07AB 584 :
00000014 07AB 585 : Output Parameters:
07AB 586 : SECRETADR = 20 ; address to return VA
07AB 587 :
07AB 588 : Implicit Outputs:
07AB 589 :
07AB 590 : Completion Codes:
07AB 591 :
07AB 592 : Side Effects:
07AB 593 :
07AB 594 :--
07AB 595 :
07FC 07AB 596 :.ENTRY EXESSYS_SECTION,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
07AD 597 :
07AD 598 : Allocate enough SPTes for image
07AD 599 :
07AD 600 :
56 0C AC D0 07AD 601 : MOVL SECPAGECNT(AP),R6
55 00000000'EF 9E 07B1 602 : DSBINT 25$ ; go to SYNCH and lock down code
57 0000'C5 D0 07BB 603 : MOVAB MMG$A_SYSPARAM,R5 ; make sure we don't use SYSGEN private copy
50 56 57 C1 07C2 604 : MOVL BOO$GL_SPTFREL-EXESA_SYSPARAM(R5),R7 ; first free SPT
0000'C5 50 D1 07C7 605 : ADDL3 R7,R6,R0 ; new free pointer
0000'C5 0A 14 07CB 606 : CMPL R0,BOO$GL_SPTFREL-EXESA_SYSPARAM(R5) ; enough left?
0000'C5 50 D0 07D0 607 : BGTR 20$ ; branch if not
0D 11 07D2 608 : MOVL R0,BOO$GL_SPTFREL-EXESA_SYSPARAM(R5) ; record the allocation
07D7 609 : ENBINT
07DA 610 : BRB 26$
07DC 611 :
07DC 612 20$: ENBINT
50 0244 8F 3C 07DF 613 : MOVZWL #SS$_VASFULL,R0
04 07E4 614 : RET
07E5 615 :
00000008 07E5 616 25$: .LONG IPL$_SYNCH
07E9 617 26$:
```



14 BC	57	09	78	07E9	618		ASHL	#9,R7,@SECRETADR(AP)	: return address of loaded code
00 14 BC	1F	E2		07EE	619		BBSS	#V\$V_SYSTEM,@SECRETADR(AP),28\$	
				07F3	620	28\$:			
				07F3	621	:			
				07F3	622	:			
				07F3	623	:			
				07F3	624	:			
7E	10	AC	DD	07F5	625		PUSHL	R7	: first SPT
7E	0C	AC	D0	07F9	626		MOVZBL	SECPROT(AP),-(SP)	: driver protection
7E	08	AC	D0	07FD	627		MOVL	SECPAGECNT(AP),-(SP)	: page count
	04	AC	DD	0801	628		MOVL	SECSTRTVBN(AP),-(SP)	: start VBN
				0804	629		PUSHL	SECCHAN(AP)	: channel
FEF1 CF	05	FB		0804	630		CALLS	#5,EXESLOAD_PAGED	
		04		0809	631		RET		
				080A	632				
				080A	633		.END		

LOADSUB  
Symbol table

## - System Code Loading Subroutines L 4

16-SEP-1984 00:28:37 VAX/VMS Macro V04-00  
5-SEP-1984 03:44:23 [SYS.SRC]LOADSUB.MAR;1Page 15  
(6)

```

$ST1          = 00000001
BADHDR        = 00000585 R      01
BOOSGL_SPTFREL ***** X      01
BOOSGL_SPTFREL ***** X      01
CCBSL_WIND    = 00000004
CHAN          = 00000004
DYN$C_DPT     = 0000001E
DYN$C_LOADCODE = 00000062
DYN$C_PAGED   = 00000002
ERRCHN       = 000006F4 R      01
ERRTYP       = 000004D5 R      01
EXESA_SYSPARAM ***** X      01
EXESGC_FLAGS ***** X      01
EXESLOAD_CODE = 00000418 RG     01
EXESLOAD_NONPAGD = 00000609 RG     01
EXESLOAD_PAGED = 000006FA RG     01
EXESSYS_SECTION = 000007AB RG     01
EXESV_SYSPAGING ***** X      01
EXESV_SYSWRTABL ***** X      01
IHDSB_HDRBLKCNT = 00000010
IOS_READVBLK  = 00000031
IOCSVERIFYCHAN ***** X      01
IPL$ASTDEL    = 00000002
IPL$SYNCH     = 00000008
ISD$C_FLAGS   = 00000008
ISD$C_VBN     = 0000000C
ISD$M_DZRO    = 00000004
ISD$M_FIXUPVEC = 00000400
ISD$M_GBL     = 00000001
ISD$M_PROTECT = 00040000
ISD$M_VECTOR  = 00020000
ISD$V_WRT     = 00000003
ISD$W_PAGCNT  = 00000002
ISD$W_SIZE    = 00000000
KW_TBL        = 00000000 R      01
MMGSALCSTX    ***** X      01
MMGSALLOCPFN ***** X      01
MMGSA_SYSPARAM ***** X      01
MMGSDALLOCPFN ***** X      01
MMG$GL_SPTBASE ***** X      01
MMG$GL_SYSPHD ***** X      01
NONPAG_ERR    = 0000058C R      01
NOSPT         = 000004C8 R      01
PAGECNT       = 0000000C
PFNSAB_STATE ***** X      01
PFNSAB_TYPE ***** X      01
PFNSAL_PTE ***** X      01
PFNSAW_REFCNT ***** X      01
PFNSC_ACTIVE  = 00000007
PFNSGC_PHYPGCNT ***** X      01
PHDSL_PSTBASOFF = 00000020
PR$ IPL       ***** X      01
PR$ TBIA      ***** X      01
PROT          = 00000010
PRT$C_ERKW    = 00000006
PRT$C_EW      = 00000005
PRT$C_KW      = 00000002
```

```

PRT$C_NA      = 00000000
PRT$C_RESERVED = 00000001
PRT$C_SREW    = 00000009
PRT$C_SRKW    = 0000000A
PRT$C_SW      = 00000008
PRT$C_UREW    = 0000000D
PRT$C_URKW    = 0000000E
PRT$C_URSW    = 0000000C
PRT$C_UW      = 00000004
PTESC_KW      = 10000000
PTESM_TYPO    = 00400000
PTESM_TYPI    = 04000000
PTESM_VALID   = 80000000
PTES$PFN      = 00000015
PTES$PROT     = 00000004
PTESV_PFN     = 00000000
PTESV_PROT    = 0000001B
READCOD       = 00000683 R      01
RETADR        = 00000008
SECSL_CCB     = 00000000
SECSL_PAGCNT  = 0000001C
SECSL_REFCNT  = 00000018
SECSL_VBN     = 00000010
SECSL_VPXPFC  = 00000008
SECSL_WINDOW  = 0000000C
SECSM_CRF     = 00000002
SECSW_FLAGS   = 00000014
SECSW_SECXFL  = 00000004
SECCHAN       = 00000004
SECPAGECNT    = 0000000C
SEC$PROT      = 00000010
SECRETADR     = 00000014
SECSTRTVBN    = 00000008
SLV$B_PROT_R  = 0000000C
SLV$B_PROT_W  = 0000000D
SLV$B_SUBTYP  = 0000000B
SLV$B_TYPE    = 0000000A
SLV$W_SIZE    = 00000008
SPT           = 00000014
SS$ BADIMGHDR = 00000044
SS$ BADPARAM  = 00000014
SS$ IN$FMEM   = 00000124
SS$ IVCHNLSEC = 0000026C
SS$ NORMAL    = 00000001
SS$ VASFULL   = 00000244
STORAGE       = 00000010 R      01
STRTVBN       = 00000008
SYSSQIOW      ***** GX     01
VASV_SYSTEM   = 0000001F
WCB$B_ACCESS  = 0000000B
WCB$C_PID     = 0000000C
WCB$V_SHRWCB  = 00000003
WCB$W_REFCNT  = 0000000E
```



+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000080A ( 2058.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.05	00:00:01.28
Command processing	120	00:00:00.51	00:00:03.67
Pass 1	408	00:00:15.12	00:00:48.36
Symbol table sort	0	00:00:02.36	00:00:07.15
Pass 2	121	00:00:03.04	00:00:08.25
Symbol table output	14	00:00:00.10	00:00:00.47
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	696	00:00:21.24	00:01:09.23

The working set limit was 1650 pages.

87480 bytes (171 pages) of virtual memory were used to buffer the intermediate code.

There were 90 pages of symbol table space allocated to hold 1555 non-local and 39 local symbols.

633 source lines were read in Pass 1, producing 27 object records in Pass 2.

31 pages of virtual memory were used to define 30 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
\$_255\$DUA28:[SYS.OBJ]LIB.MLB;1	15
\$_255\$DUA28:[SYS.LIB]STARLET.MLB;2	12
TOTALS (all libraries)	27

1718 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:LOADSUB/OBJ=OBJ\$:LOADSUB MSRC\$:LOADSUB/UPDATE=(ENH\$:LOADSUB)+EXECML\$/LIB



0377

AH-BT13A-SE  
 VAX/VMS V4.0

**DIGITAL  
CONFIDE**

EQUIPMENT  
INITIAL AND

CORPORATION  
PROPRIETARY